

Conceptual graphs for the analysis and generation of sentences

by Paola Velardi
Maria Teresa Pazienza
Mario De' Giovanetti

A system for analyzing and generating Italian texts is under development at the IBM Rome Scientific Center. Detailed semantic knowledge on word-sense patterns is used to relate the linguistic structure of a sentence to a conceptual representation (a *conceptual graph*). Conceptual graphs are stored in a database and accessed by a natural-language query/answering module. The system analyzes a text supplied by a press-agency-release database. It consists of three modules: a morphological, a syntactic, and a semantic processor. The semantic analyzer uses a *conceptual lexicon* of word-sense descriptions, currently including about 850 entries. A description is an extended case frame providing the *surface semantic patterns* (SSP) of a word-sense *w*. SSPs express both semantic constraints and word-usage information, such as commonly found word patterns, idioms, and metaphoric expressions. SSPs are used by the semantic interpreter to build a conceptual graph of the sentence, which is then accessed by the query-answering and language-generation modules. This paper makes the claim that the

SSP approach is viable and necessary to cope with language phenomena in unrestricted domains. Surface patterns are easily acquired inductively from the natural-language corpus rather than deductively from predefined conceptual structures. SSPs map quite complex sentences into surface semantic representations that can be generalized at a subsequent stage. In contrast, the current state of the art does not provide viable theory or methodology to go from superficial to deep structures. This issue is more extensively addressed in the body of the paper.

1. Introduction

A natural-language-processing (NLP) system is being developed at the IBM Rome Scientific Center. The system takes as input Italian sentences and produces a representation of their meaning. A prototype has been implemented to analyze a database of press-agency releases on finance and economics.

This paper outlines the method by which the system performs the analysis and generation tasks, with emphasis on the semantic processor. A framework is presented to encode and use semantic knowledge for analysis and synthesis of natural-language texts. A diagram of the system is given in **Figure 1**. Generation features are used primarily for question answering, but are also employed to paraphrase the input sentences. The conceptual representation of the sentence is a directed graph of *concepts* and *conceptual relations*, called a conceptual graph (CG). This formalism, introduced in [1], is

©Copyright 1988 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the *Journal* reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free without further permission by computer-based and other information-service systems. Permission to *republish* any other portion of this paper must be obtained from the Editor.

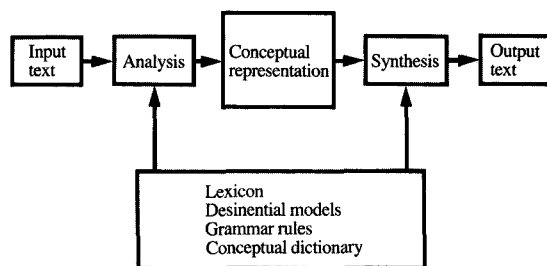


Figure 1

Scheme of the text analysis/synthesis process.

a versatile and powerful representation model, derived from the well-known concept of *semantic nets*.

In our work we have made an effort toward implementing a rich semantic knowledge base. Many authors of NLP papers give special emphasis to issues of semantic knowledge representation, but at the same time appear to be skeptical about the viability of a strong semantic approach. A frequent argument is that the implementation of (even a partial) world-knowledge component would be time-consuming, and its practical application prohibitive [2]. An opposing claim is made by the pioneers of the *Knowledge Principle* ("... if a program is to perform a complex task well, it must know a great deal about the world in which it operates ...") [3], who stress the importance of large knowledge bases as a starting point for any working AI system.

Despite this interest, it appears that no extended efforts have been made to encode lexical knowledge on a systematic basis. Some work in this direction has recently been reported in [4–6]. Reference [4] presents a system that acquires semantic information by analyzing on-line dictionary entries with the help of a fast and efficient syntactic analyzer [7]. The system has some limitations but shows that semantic information can in part be acquired automatically. In [6], commonsense knowledge about verb uses has been introduced manually using psycholinguistic data. In [8], an in-depth analysis of language-acquisition issues is provided, along with a program, RINA, which is able to learn new word patterns and idioms using a lexicon of *pattern-concept-presupposition* (PCP) triples. Two important conclusions emerge from this paper: 1) Idioms, figures of speech, and metonymy are so common in language that an NL computer program cannot simply ignore them. 2) Language acquisition at a sizable level must rely on a *uniform set of learning strategies*.

In this paper, our main objective is to prove the viability of a large and thoroughly defined *semantic lexicon*, which we

believe is an obligatory choice in order to cope with unrestricted natural-language domains. In our lexicon, extended case frames are stored for each word sense. Word-sense definitions express semantic constraints as well as commonly used word patterns, idiomatic expressions, and metaphorical extensions [e.g., *viaggiare in cattive acque* — literally, "to travel in bad waters," meaning 'to be in trouble' (English equivalent: 'to be in hot water'); *dare a piene mani* — literally, "to give with full hands," i.e., 'to give wholeheartedly']. In this paper, these are referred to as the *surface semantic patterns* (SSPs).

The knowledge-representation model presented hereafter does not claim to be language-independent; the final representation of a sentence is *micro-semantic* [9], i.e., tied to the surface sentence. We believe that the ultimate goal of a language-understanding system is to produce a "deep" representation, but the methods by which this representation should be derived are unclear and not generally accepted in the present state of the art.

As a first stage in the development of the system, we have provided it with the ability to derive a surface semantic representation even for complex sentence structures and to answer queries about them, as shown by the examples throughout the paper. This ability is due to, and limited by, the world knowledge represented within the semantic lexicon (currently about 850 extended word-sense definitions). It seemed reasonable to start with a small but carefully defined world model and then expand it; in fact, the insertion of new entries neither requires changes in the structure of the system nor affects processing time.

The SSP approach simplifies the acquisition of lexical knowledge: Word patterns are acquired inductively by looking at the examples in the NL corpus, rather than inferred from predefined conceptual structures. In our group, a *knowledge editor* has been developed to simplify the entry of word definitions; a project to develop a tool for word-pattern acquisition and generalization was recently started. As in [4], this tool makes use of the syntactic parser developed for our text analyzer [10, 11].

In passing, we would like to point out that the contribution of this work is not in the field of linguistics, but rather in the field of knowledge engineering. The actual content of the semantic knowledge base (of which we provide some examples) could surely be improved by linguistic experts.

The succeeding sections present some of the features of the system. The next section briefly overviews the basic components of the text processor. The objective of this section is to describe in detail the knowledge-representation model and to show how SSPs are used by the system to derive a semantic interpretation of a sentence previously analyzed by a morphological and syntactic processor. The result of semantic analysis is a formal representation of the input text, called a *conceptual graph* [1]. The conceptual

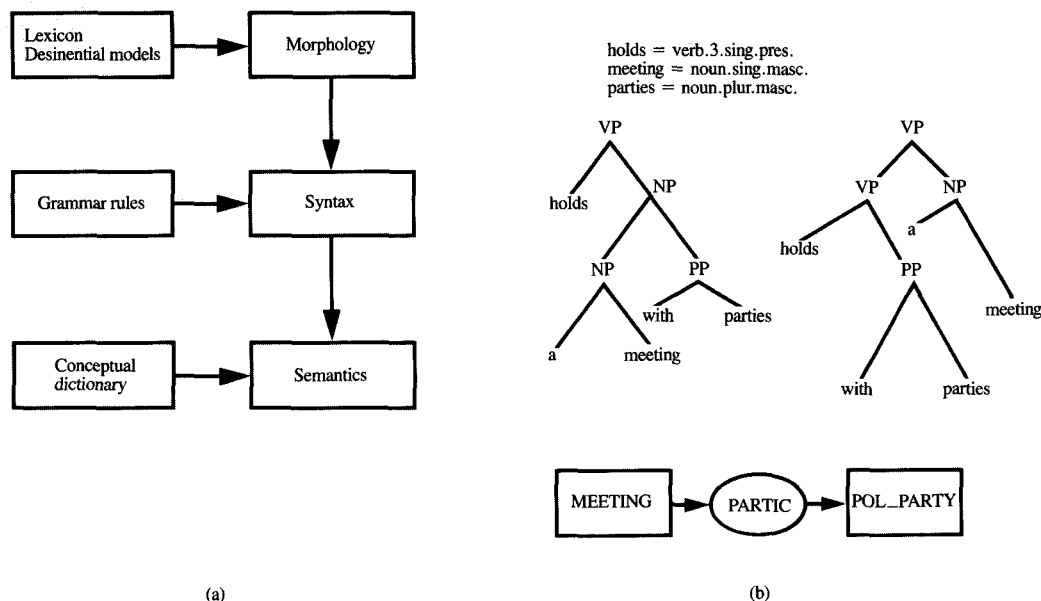


Figure 2

Text analyzer: (a) schematic; (b) sample output.

graphs of the analyzed texts are stored in a database and retrieved by natural-language queries. Query processing and answer generation are discussed in Section 3. That section investigates the ability of the system to "understand" a piece of text and gives some insight into the power and limitations of the semantic representation model we have adopted.

More detailed descriptions of the semantic knowledge base and text analysis are given in [12] and in [13, 14], respectively. The other system components (morphology and syntax) are discussed in [10, 11, 15, 16].

2. Overview of the text analyzer

The NLP system is based on three levels of analysis: morphology, syntax, and semantics. A diagram of the text analyzer, together with some sample output, is presented in Figure 2. Text synthesis is also performed in three steps. Many systems described in the literature, such as those following the theories of Schank and Wilks, build semantic structure directly from the input string. Other systems, e.g., the well-known SHRDLU [17], are based on an interleaved processing of syntax and semantics. PHRAN and PHRED [18, 19] use a data structure where syntactic and semantic knowledge are stored in single units called *pattern-concept* pairs.

In our system there is some interleaving of syntactic and semantic processing [11], but semantic, syntactic, and morphological knowledge are kept separate. According to a definition given in [20], the system is *sentence final*; i.e., a first pass builds the syntactic structure and the second a semantic representation. Indeed, we have found a considerable advantage in this separation, which results in a cleaner and more systematic representation scheme, and (we believe) has simplified the analysis and synthesis algorithms.

The morphology, described in [15] and [16], associates at least one stem with each word; in Italian this task is particularly complex because of recursive generation mechanisms, such as derived forms, verbs used as nouns, etc. For example, from the stem *casa* ('home') can be derived the words *cas-etta* ('little home'), *cas-ett-ina* ('nice little home'), *cas-ett-in-accia* ('ugly nice little home'), and so on. Forms are generated using a context-free grammar; in this grammar, the word is the root form, and prefixes, suffixes, derived forms, endings, and enclitics are the terminal elements. At present, the system performs morphological analysis of about 100 000 words (100% coverage of the analyzed corpus), using a lexicon of 7000 *elementary lemmata* (stems without affixes). The morphological component can both analyze and generate words.

The syntactic analyzer [10] determines syntactic relationships among words by verifying grammar rules and agreement of forms. The system is based on an attribute grammar augmented with *lookahead sets* (first introduced in formal languages [21]) and local semantic checks [11]. The grammar has more than 100 rules which provide wide coverage of the corpus analyzed (roughly 80%). The rules account for such language phenomena as multiple coordinate constructions, multiple prepositional phrases which have several possible places to be attached, subordinate and parenthesized clauses, interrogatives, etc.

The style of Italian written text is quite complex; in fact, sentences are usually composed by nested hypotactical phrases, rather than linked paratactically. For example, consider the sentence¹

Il Primo Ministro si è recato con un gruppo di esperti alla centrale nucleare di Latina ad incontrare dei delegati cittadini per un dibattito sulla politica nucleare del governo.

"The Prime Minister went with a team of experts to the nuclear plant of Latina to meet city delegates for a meeting on the nuclear policy of the government."

'The Prime Minister visited the nuclear power plant at Latina with a team of experts and debated with a citizen delegation about the government's nuclear policy.'

The above example shows that prepositional attachment is a major problem. Syntax only reveals the surface structure of a sentence; it is the task of semantics to make explicit the nature of links between words. For example, in the above sentence about the Prime Minister's visit, the preposition *con* ('with') means *accompaniment* ("Minister with a team of experts"). The first occurrence of the preposition *a* (*a-lla* = 'to the') indicates the *destination* of the act *go* ("went to the plant"), the second occurrence (*a-d*) indicates the *purpose* of the trip ("to meet delegates"), etc.

The conceptual graph formalism [1], developed by John Sowa as a modification and extension of semantic nets, seems particularly suitable for capturing semantic relations between words; according to this model, words are represented by *concepts* and related to one another by *conceptual relations*. For example, the sentence *The Ace signs a contract* has the graph

[COMPANY: Ace] ← (AGNT) ← [SIGN] → (OBJ) → [CONTRACT]

This graph is presented in what is called the *linear form* of conceptual graphs; Figure 2 uses the equivalent graphic form.

A concept might refer to a generic instance ("a company"), to a specific individual ("the company" or

"Ace"), or to a set of generic ("companies") or specific ("the Ace, the Cea, and the Eca Companies") instances. This information is given by the *concept referent*, respectively, as follows:

[COMPANY: *] or simply [COMPANY]

[COMPANY: #] or [COMPANY :Ace]

[COMPANY: {*}]

[COMPANY: {Ace Cea Eca}]

The symbols *, #, and {} represent generic instance, specific instance, and set of instances, respectively.

Concepts are ordered in a *type hierarchy*; for example,

HUMAN > HUMAN_ORGANIZATION > COMPANY.

The following subsections describe how conceptual graphs have been used in our system to implement a semantic analyzer.

The semantic knowledge base

In order to represent a sentence by a conceptual graph, the system is provided with a *semantic lexicon* of word-sense descriptions. This section describes the structure of the semantic knowledge base.

Many NLP systems express semantic knowledge in the form of *selectional restrictions* or *deep case constraints*. In the first case, semantic expectations are associated with the words employed, as in *canonical graphs* [22]; in the second case, they are associated with some abstraction of a word, as in Wilks' *formulas* [23] and Schank's primitive *conceptual cases* [24].

Semantic expectations, however, do not provide enough knowledge to analyze and resolve many language phenomena; as noted in [25], they refer to the conceptualization underlying a sentence rather than to its linguistic structure. Semantic expectations are useful for mapping a surface structure into a deeper one, but often are not helpful in the interpretation of complex language patterns, which are better understood by using world knowledge.

In the following, some examples are provided for which world knowledge is necessary:

1. *Metonymy*

La Regione, la Ace e il sindacato firmano un accordo.

'The local government, Ace, and the trade union sign an agreement.'

L'incontro si è tenuto alla Ace di Roma.

'The meeting was held at Ace of Rome.'

'The meeting was held at the head office of Ace of Rome.'

In the first sentence, *Ace* designates a human

¹ In order to show the nature of the problems encountered, all the examples which are not literally translatable from Italian to English are provided in three versions: Italian, a word-by-word English translation, and idiomatic English.

[PROJECT] ← (KIND_OF) ← [MENTAL_ACT]

[PROJECT]—

- (ORIGINATOR) → [HUMAN] (e.g., the project *of* the Ace Company)
- (PURPOSE) → [ACTIVITY] (e.g., a project *for* an increase of productivity)
- (OBJ) → [CONCRETE] (e.g., the project *of* new buildings)
- ← (RESULT) ← [MENTAL_ACT] (e.g., to design a project)
- ← (ARGUMENT) ← [COMMUNICATION_ACT] (e.g., to propose a new project)
- (MANNER) → [ACT_ATTRIBUTE] (e.g., an effective project)

Definition of the word-sense PROJECT.

organization: It is some delegate of the Ace Company who actually signs the agreement. In the second sentence, *Ace* designates a plant or the head office where a meeting took place.

2. Syntactic ambiguity

Il Primo Ministro White si è recato a Milano per un convegno.

'Prime Minister White went to Milan for a meeting.'

Il Presidente Brown si è recato in un centro per handicappati.

'President Brown visited a residence for the handicapped.'

In the first case, "meeting" is the purpose of the verb *go*; in the second case, "handicapped" refers to the purpose of a building or institution (the noun *centro*). In both examples, syntactic rules are unable to determine whether the prepositional phrase should be attached to the noun or to the verb. Semantic expectations in general provide selectional restrictions only for strongly expected concept modifiers; thus, the restrictions attached to the act GO (see, for example, [1]) impose constraints only on the *agent* and the *destination*, but do not say anything about the semantic validity of a *purpose* modifier.

3. Coordinate constructions

un incontro tra sindacalisti, il Ministro degli Interni, Doe e la Regione

'a meeting between trade unionists, the Minister of the Interior, Doe and the local government'

The first comma links two different human entities; the second indicates that what follows (i.e., Doe) is an appositive.

The above phenomena, plus many others, such as metaphors, vagueness, and ill-formed sentences, can be dealt with by representing *world knowledge* in the semantic knowledge base. Knowledge about pragmatic word usages, contexts, figures of speech is potentially unlimited, but it enables the handling of natural-language texts without severe restrictions.

To define a semantic lexicon we have adopted the following guidelines:

1. Each word-sense has an entry in the semantic lexicon, called a *concept definition*.
2. A concept definition is an extended case frame describing semantic expectations *and* semantically permitted *uses*. For example, a *purpose* and a *manner* are indicated as possible modifiers for *go*; figures of speech are also included, as in *The car drinks gasoline*.
3. Each word use is represented by an *elementary graph*, expressing a *surface semantic pattern* (SSP):

(1) [W] ↔ (CONC_REL) ↔ [C]

where W is the concept to be defined, C a concept type, and ↔ is either a left- or a right-pointing arrow. The concept C is the most general type in the hierarchy for which (1) holds.

An example of word-sense definition is given in **Figure 3**; the first graph defines the *supertype* of PROJECT in the type hierarchy. The other elementary graphs are the SSPs in which the word-sense PROJECT occurs. Each graph is

SENTENCE $\leftarrow$ NP + VP
 NP $\leftarrow$ art + noun | noun | adjective + NP | NP + PP
 VP $\leftarrow$ verb | VP + NP | VP + PP
 PP $\leftarrow$ preposition + NP | preposition + VP

Figure 4

A simple grammar.

provided with a natural-language proposition or phrase that would be represented by that graph. For example, the first reads "PROJECTs are originated by HUMANS"; the phrase "the project of the Ace Company" is a specialization of this graph (Ace is a COMPANY and COMPANY can have the HUMAN attribute).

Conceptual relations are of the type proposed by [26]; the set of relations (about 50) is much wider than in [26] and [1] to account for all the basic relations found. A more detailed example of concept definition is given in the Appendix. For a discussion of conceptual-relation types and their relations with surface linguistic structures, see [12].

An important problem with this definition scheme is the "completeness" of word-sense definitions. It is *not* a goal of our knowledge-representation framework to seek the most general "meaning" of a word. We believe the solution to this problem, if a solution exists, is, rather, a task for philosophy. In contrast, our objective is to *describe* how a word-sense is used in language rather than to express the mental model behind that word. This approach has at least one advantage: If a new aspect of a word is found, it is simply added to the word definition; whereas if a conceptual model proves to be inadequate (for example, if its conceptual primitives are not adequate to capture some world aspect), almost everything in the system must be reconsidered.

At present, the system has some 850 definitions, each including about 10–20 SSPs encoded with the help of a *concept editor*. SSPs are acquired inductively, by looking at all the occurrences of a given word in contexts; contexts are available on-line. Many SSPs have been found common to a conceptual category and encoded in a standard form. For example, all MOVE_ACTs have a DEST(ination) and a SOURCE among their SSPs. Similarities among word patterns are exploited using the relations of near-synonymy and antonymy; for example, the acts CLOSE and OPEN have an identical behavior as far as their use in language is concerned. Near-synonymy and antonymy relations are used to increase the size of the lexicon, even though they might produce some error. The coverage vs. accuracy trade-off in using near-synonymy relations for undefined words is being investigated.

The concept editor performs a variety of tests on newly defined SSPs. For example, an SSP including the conceptual relation CONC_REL must satisfy the semantic constraints imposed by that relation. A second facility provided by the editor is the detection of similarities between SSPs of different word-senses, as well as "suspected" inconsistencies. A detailed description of these features, however, is outside the scope of this paper.

The semantic interpretation algorithm

This section describes how SSPs are used by the semantic interpreter to derive a conceptual graph of a sentence.

The input to the system is provided by the syntactic parser, based on an attribute grammar. The purpose of syntactic analysis is to detect the possible relations between words; in fact, SSPs only provide information on valid *concept pairs*, such as

[PROJECT] $\rightarrow$ (ORIGINATOR) $\rightarrow$ [HUMAN]

but give no guidance on how to detect these pairs within a sentence.

For a brief explanation of the interface between syntax and semantics, let us consider the simplified grammar shown in Figure 4. The rules appearing there do not show, for the sake of brevity, the conditions on the attributes (grammatical agreement, attribute inheritance, and look-ahead sets); the interested reader may refer to the papers mentioned in Section 2 for a detailed description of the syntactic parser.

In general, a sentence has more than one syntactically valid interpretation and gives rise to several *parse trees*. For example, the verb phrase *to discuss the proposal of a plan* produces the parse trees of Figure 5.

Parse trees are internally represented in logical form by *syntactic predicates* (SPs). An SP has the form

syntactic_predicate_name(*x,*y).

where *x and *y are either words or pointers to an NP or VP. Figure 6 shows the correspondence between the grammar of Figure 4 and the SPs.

Applying these correspondence rules, for example, to the parses of the sentence

Gli azionisti in assemblea discutono la proposta di un piano per recuperare l'efficienza delle strutture produttive.

"The shareholders in assembly discuss the proposal of a plan to restore the efficiency of productive structures."

'The shareholders in a meeting discuss a project to increase the productivity of the firm.'

we obtain the following SPs:

1. NP_PP(shareholder,in,assembly).
2. ATTRIBUTE(structure,productive).

The semantic analyzer first examines each SP separately. Consider, for example, the NP_PP(proposal,of,plan). The first step is to replace stems with word-senses. It is interesting

1. Associate a unique word-sense with each word.
2. Replace syntactic links between words with semantic (conceptual) relations between concepts.
3. Create a conceptual graph of the input sentence.

Its purpose is to

the emphasis is on the noun, or partitive constructions such as the majority of the delegates, where the head is delegates. Semantic verification is an important part of the analysis.

Correspondence between grammar rules and syntactic predicates.

Figure 6

SENTENCE → SUBJ(*vp,*np)
 VP + NP → OBJ(*vp,*np)
 adjective + NP → ATTRIBUTE(*np,*adj)
 NP + PP → NP_PP(*np,*prep,*pp)
 VP + PP → VP_PP(*vp,*prep,*pp)

Hence, for example, the head of SP 1 is *shareholder*, the head of SP 2 is *structure*, etc. The conversion of syntactic links between phrases into binary relations between words is admittedly a simplification; it is based on the intuitive argument that in a noun (verb) phrase it is the noun (verb) that conveys most of the semantic information. Thus, a semantic relation between phrases mostly depends upon the relation between their heads. Exceptions to this rule include relative clauses such as in *the contract that was signed*, where

head(NP) = noun,
 head(VP) = verb.

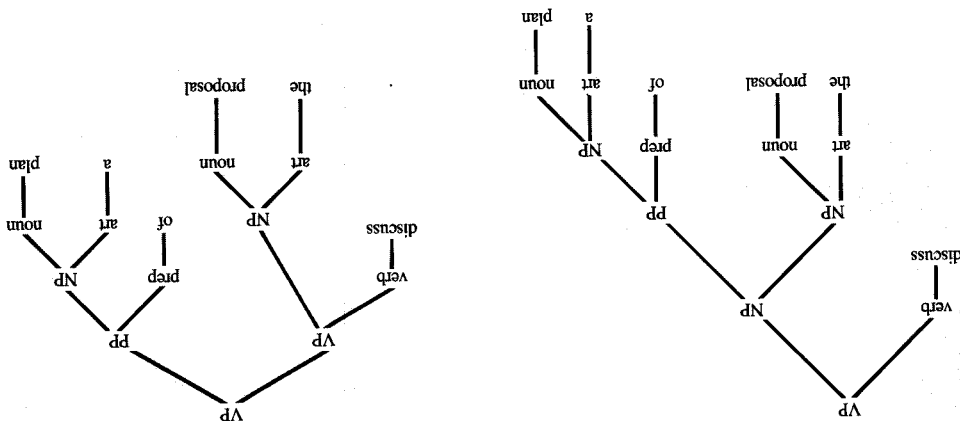
follows:

The arguments of a syntactic predicate are either a word or a pointer to some other SP. In the preceding example, morphological information attached to each word is omitted for brevity; only the stem is shown. It should be noted that the example is a representation of only one of the valid parse trees derived by the system. During semantic analysis, SPs are analyzed in the order indicated in the example. A *head* word is associated with each predicate; the head is in most cases determined as

3. NP_PP(afficiency,of,2).
4. OBJ(restore,3).
5. NP_PP(plan,to,4).
6. NP_PP(proposal,of,5).
7. OBJ(discuss,6).
8. SUBJ(7,1).

Parse trees for the phrase *discuss the proposal of a plan*.

Figure 5



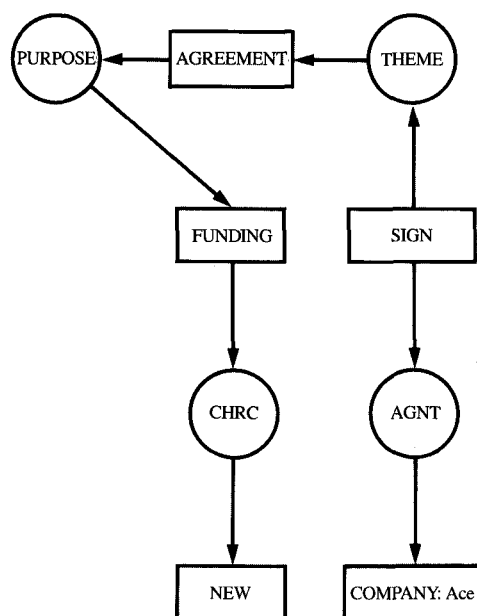


Figure 7

Example of conceptual graph generated by the Italian text analyzer.

to note that the Italian word *piano* is particularly ambiguous: As a noun, it means 'plan' (as in the example), 'piano,' 'project,' 'floor,' and 'level land.' When this word is first encountered in a syntactic predicate, a list of all its word-senses is associated with it. Hence the SP is rewritten in four different ways:

NP_PP(PROPOSAL,of,PIANO),
 NP_PP(PROPOSAL,of,PROJECT),
 NP_PP(PROPOSAL,of,FLOOR),
 NP_PP(PROPOSAL,of,LEVEL_LAND).

In step 2, each of the above SPs is verified for semantic plausibility. The test is performed by first associating with syntactic predicates a list of conceptual relations. A conversion table called SINT_SEM performs this task: For example, an NP_PP with the preposition "of" can be rewritten

NP_PP(*x,of,*y) ← POSSESS(*y,*x). (e.g., *the book of Bill*),
 NP_PP(*x,of,*y) ← PART_OF(*y,*x). (e.g., *the pages of the book*),
 NP_PP(*x,of,*y) ← ARGUMENT(*x,*y). (e.g., *the book of history*),
 etc.

The right-hand side of each rule, called its *semantic hypothesis* (SH), represents a possible interpretation of an SP; for example, the first rule is verified if 'the project possesses a proposal' is a plausible interpretation of the NP_PP *the proposal of a plan*, as it would be for *the book of Bill* (i.e., 'Bill's book').

The semantic hypothesis
 POSSESS(PROPOSAL,PROJECT) is *plausible* iff the corresponding graph

[PROJECT] ← (POSSESS) ← [PROPOSAL]

is the specialization of, or matches with, some SSP included in the concept definition of PROPOSAL and of PROJECT.

The SH of the above example cannot be proved and is hence rejected. The algorithm then attempts to verify a new SH, until either a plausible relation is found or no more SHs are available for that given syntactic predicate. In the latter case, the syntactic predicate is rejected and the system backtracks to the next one (if any). In the example above, the relation ARGUMENT is finally taken as a valid one. In fact, the definition of PROJECT (cf. Figure 3) includes

[PROJECT] ← (ARGUMENT) ←
 [COMMUNICATION_ACT]

which is a generalization of

[PROJECT] ← (ARGUMENT) ← [PROPOSAL]

given that

[PROPOSAL] ← (KIND_OF) ←
 [COMMUNICATION_ACT]

The analysis proceeds in the same way for all the SPs. If an ambiguous word appears in more than one SP, the subsequent analyses consider only the word senses which have not been rejected by the preceding tests. If more than one parse has been generated, the analysis applies to all the parse trees. In case of genuine ambiguity (e.g., *I watched a man with binoculars*), word proximity is used to select an interpretation. A better approach would be to consider contextual information, but at present the system does not have discourse-analysis capabilities. Moreover, in our natural corpus of short narrative texts, all the ambiguities can be resolved.

At the end of the analysis, provided that at least one interpretation is found, the system outputs a conceptual graph of the sentence. Figure 7 shows the graph of the sentence considered in this example. The figure illustrates the graph actually generated by the system graphic facility (Italian labels have been replaced by English labels). Within the limits imposed by the size of the lexicon, the system is able to analyze complex sentences (more than twenty words and five or six prepositional phrases), previously processed by the morphological and syntactic analyzers. All the sentences used as examples throughout this paper are successfully analyzed by the system.

The conceptual graph of a sentence is a *surface* semantic representation, because it is derived by associating an SSP to each syntactic predicate. The graph is still tied to the sentence clauses, but the words are disambiguated and the syntactic relations and functional morphemes (conjunctions, prepositions, etc.) are replaced by conceptual relations. For example, constructions like

The delegates agree on a project.

the agreement of the delegates about a project

Doe nominated Brown to the board of directors.

Brown was nominated by Doe to the board of directors.

Doe and Brown were elected to the board of directors.

Doe was elected with Brown to the board of directors.

have the same graph. However, the sentences

Doe and Brown were elected to the board of directors.

The board of directors nominated Doe and Brown to be members.

Doe and Brown have been on the board of directors since yesterday.

have slightly different graphs.

One research objective is to provide the system with the ability of detecting *analogies* among "similar" sentences and to answer queries about facts which are not explicit in a sentence. But again, this ability should be given by *describing* presuppositions, causes, results, and purposes commonly associated with word-senses (or word categories, when possible), rather than by decomposing concepts into primitives. Methods based on word decomposition, with and without primitives, pose the same philosophical issues as conceptual lexicons: How do we get to the "real meaning" of things? When should we stop a decomposition or deductive process?

On the other hand, knowledge of surface semantic patterns is "deep" enough to provide word-sense disambiguation and to discover meaning relations among word-senses. The following section, concerned with query answering and language generation, gives more precise insight into the power and limitations of this representation.

3. The query-answering and language-generation modules

The system is provided with a query-answering module, whose structure is given in Figure 8. Queries to the database are made in natural language and processed in four steps:

1. NL queries are analyzed by the NL processor and turned into a conceptual graph (*query graph*) with one or more uninstantiated concepts.
2. A match algorithm retrieves the graph(s) in the database that most closely relate to the query.

3. An answer is created (*answer graph*) by entailing part of the retrieved graph(s).
4. The answer is processed by a language-generation module to produce an utterance.

The main objective is to test the system's ability to provide some level of understanding of a text, i.e., to verify the depth of the information conveyed by a surface conceptual graph. An analysis of the query-answering module also gives some insight into the power and limitations of our surface semantic representation.

In this section the language-generation process is also described. This is used both for producing an NL answer and for paraphrasing an input text. As stated in [27] and [28], language generation consists of a *planning* process and a *tactical* component. The first component is concerned with such problems as deciding which words should be used to represent a semantic structure [29], what are the goals of the utterance (e.g., answering a query, translating a text, producing an explanation), *what to say* at a given step, etc. The tactical component is concerned with the linguistic processing, i.e., *how to say* in a target language the message built in the preceding phase.

In our system we have restricted ourselves to the second phase; in fact, both paraphrasing and answering processes deal with single (even though complex) sentences, represented by surface graphs. This greatly simplifies or eliminates many planning problems.

• Semantic analysis of a query

To describe the query-processing phase, let us consider the following examples:

What is the purpose of the agreement?

Why is the agreement signed?

What does Ace do?

First, the NL query is translated into a conceptual graph. The query-processing module has some additional features with respect to the semantic-interpretation algorithm presented in the preceding section. Expressions such as *What is the purpose of X*, *What is the reason for X*, *What is the argument of X*, etc., are handled by attaching the relations PURPOSE(X,?) or ARGUMENT(X,?) to the conceptual graph of X. Similarly, in *where*, *when*, and *why* questions, the interrogatives are replaced by PLACE, TIME, and PURPOSE relations, respectively. Generic verbs like *do* and *make* are handled by replacing the concept representing the verb with a question mark. For example, the third query of the preceding example reads *What is the action performed by Ace?* This manipulation of the input is performed after syntactic analysis and before the standard semantic verification.

A query might give rise to more than one query graph, due to uninstantiated concepts. For example, the third query

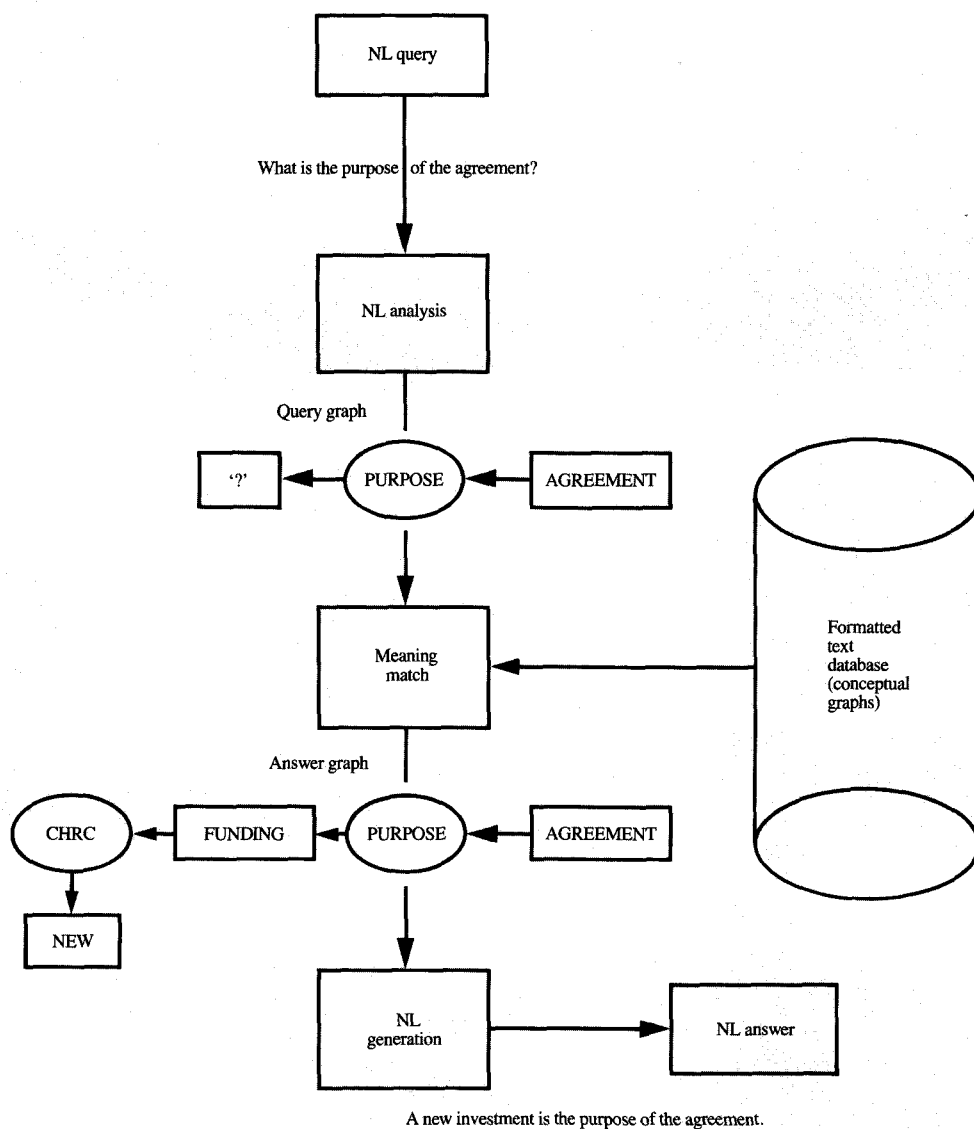


Figure 8

Scheme of the query-answering system.

after the manipulation just described is represented by the syntactic predicate

SUBJ(?,Ace).

The SP SUBJ(*x,*y) semantically corresponds to the following conceptual relations:

SUBJ(*x,*y) ← AGNT(*x,*y). (e.g., *John eats an apple*),

SUBJ(*x,*y) ← PARTICIPANT(*x,*y). (e.g., *John flies to New York*),

SUBJ(*x,*y) ← EXPERIENCER(*x,*y). (e.g., *John is worried*),

SUBJ(*x,*y) ← INSTRUMENT(*x,*y). (e.g., *The key opens the door*),

SUBJ(*x,*y) ← CAUSE(*x,*y). (e.g., *The rain wets the earth*),

SUBJ(*x,*y) ← P.SUBJ(*x,*y). (e.g., *The ball rolls*),

etc.

Among these relations, the first two apply to a *company*; the lack of additional semantic constraints imposed by the verb does not allow the selection of a unique interpretation.

Hence, after step 1, two graphs are generated:

[COMPANY: Ace] ← (AGNT) ← [?]

[COMPANY: Ace] ← (PARTICIPANT) ← [?]

• Matching and entailment of answers

The match algorithm finds for each *query graph* QG, all the graphs CG_i in the Formatted Text Data Base (FTDB), such that there exists a *projection* PQG_j of QG_i on CG_j. The *projection* operation² is here intended as in [1], with the extension that for each concept *c* of QG_i there must be a concept *pc* of PCG_j such that either *c* = *pc* or *c* and *pc* are what is called in Italian "paronimi."³

The projection operation causes the uninstantiated concepts of QG_i, *c₁, ..., *c_n to be unified with the correspondent concepts of PCG_j. Let x₁, ..., x_n be the instantiation in PQG_j of *c₁, ..., *c_n in QG_i. The *answer graph* AG is obtained by adding to PQG_j all the modifiers of x₁, ..., x_n which are not included in PQG_j but are included in CG_j. For example, the QG AGNT(?company(Ace)) matches with the graph

[COMPANY: Ace] ← (AGNT) ← [SIGN]

→ (THEME) → [AGREEMENT]

→ (PURPOSE) → [FUNDING] → (CHRC) → [NEW]

The PQG is

[COMPANY: Ace] ← (AGNT) ← [SIGN]

hence x₁ = SIGN; the modifier of x₁ not included in PQG is

→ (THEME) → [AGREEMENT]

The AG will be

[COMPANY: Ace] ← (AGNT) ← [SIGN] → (THEME) → [AGREEMENT]

If more than one match occurs, more than one answer to the initial query is produced. Multiple answers are also generated if more than one QG of the initial query has a match in the FTDB.

One thing should be stressed at this point: The match between graphs is simply a projection algorithm. If the query

and the corresponding graph do not refer explicitly to the same facts, the match does not succeed. For example, given the sentence *Doe and Brown were nominated yesterday to the board of directors of Ace*, no match would be found for queries like *Who is now in Ace?* In order to answer that query, knowledge about *presuppositions* and *results* associated with events must be encoded in the semantic lexicon. Nevertheless, if a naive user is asked to pose questions concerning an analyzed text, an answer is correctly produced for the majority of cases; we believe this to be an encouraging result. For example, given the analyzed sentence (in English translation)

"During a meeting for new funding, President Doe nominated yesterday at Rome Mr. A, Mr. B and Mr. C to be members of the board of directors."

the following is a list of queries the system *can* answer (the answer is also shown, as given by the system during a query session):

Q. *Chi nomina il presidente Doe?* ("Who nominates the President Doe?" i.e., "Who does President Doe nominate?")

A. *Il Presidente Doe ha nominato membri Mr. A, Mr. B e Mr. C.* ('President Doe nominated Mr. A, Mr. B and Mr. C to be members.')

Q. *Chi nomina Mr. B membro del consiglio di amministrazione?* ('Who nominates Mr. B to be a member of the board of directors?')

A. *Il presidente Doe ha nominato Mr. B membro del consiglio di amministrazione.* ('President Doe nominated Mr. B to be a member of the board of directors.')

Q. *Quando ha nominato Mr. C il presidente?* ('When did the President nominate Mr. C?')

A. *Il presidente ha nominato Mr. C ieri.* ('The President nominated Mr. C yesterday.')

Q. *Dove sono stati nominati Mr. C e Mr. D?* ('Where were Mr. C and Mr. D nominated?')

A. *Mr. C e Mr. D sono stati nominati a Roma.* ('Mr. C and Mr. D were nominated at Rome.')

Q. *Cosa ha fatto ieri il presidente?* ('What did the President do yesterday?')

A. *Il presidente ha nominato a Roma ieri in un incontro Mr. A, Mr. B e Mr. C.* ('The President nominated Mr. A, Mr. B and Mr. C at Rome yesterday in a meeting.')

Q. *Quale è lo scopo dell'incontro?* ('What is the purpose of the meeting?')

² Given two graphs *u* and *v*, a projection *p*: *v* → *u* is a graph *pv* such that for each concept *c* in *v* there is a concept *pc* in *pv* where *pc* ≤ *c* in the hierarchy, and for each relation REL in *v* there is a relation P_REL in *pv* where REL = P_REL. If the argument *i* of REL is *c*, then the argument *i* of P_REL is *pc*.

³ There appears to be no commonly accepted equivalent term in English. The concept is that of near-synonymy (as, e.g., in the English words *company* and *firm*).

A. *I nuovi fondi sono lo scopo dell'incontro.* ('The purpose of the meeting is new funding.')

Answer graphs are the input to the generation module, described in the next section.

• The language-generation module

When a text is analyzed, its linguistic structure is data for the analysis; the objective is to explicate the *conceptual model* behind the text. In language generation, the problem is reversed: The *meaning* is a given of the problem; the linguistic structure is to be found. To create an utterance out of a purely semantic representation, the system makes several decisions:

1. Active versus passive form. The graph

[COMPANY: Ace] ← (AGNT) ← [SIGN] → (THEME)
→ [CONTRACT]

may be expressed by the utterances *Ace signs a contract* or *A contract is signed by Ace*.

2. Synthetic versus direct replacement. The graph

[DELEGATE: {#}] ← (PARTICIPANT) ← [MEETING]

can be read *a meeting between delegates* or *The delegates participate in a meeting*. In the first case, the conceptual relation is expressed by a preposition (*between*); in the second case, it is replaced by a verb complex (e.g., [*to participate in*]).

3. Emphasis. The graph

[ITEM: {#}] ← (PART_OF) ← [STATUTE]

can be read *the items of a statute* or *a statute with items*, depending upon the focus of the sentence including the phrase.

4. Ordering. The graph

[IMPORTANT] ← (CHRC) ← [AGREEMENT]

is read in English *an important agreement*; in Italian, however, the adjective can also be postposed: *un accordo importante*.

5. Synonyms. Consider the graph

[POL_PARTY: #] ← (ORIGINATOR) ← [BELIEF]

As discussed in Section 2, semantic ambiguity of words is resolved during the analysis process: In this case, POL_PARTY indicates a political party, rather than party in the sense of celebration. However, a word-sense might correspond to many words (synonyms and near-synonyms); for example, this graph can be read *the*

belief of the party, the opinion of the party, or the view of the party.

Issues of these types are faced during the appropriate steps of the synthesis process; in some cases, straightforward solutions are adopted.

It is worth remembering that the starting point of our generation module is a surface graph which can be uttered in one sentence. Many problems, such as the selection of the appropriate words [29], level of detail, and focus [30], have a minor influence in this context.

The generation process consists of three phases, NEST, GEN_TREE, and MORPH. Each of these modules is described in detail in the following subsections.

Nesting a conceptual graph

The purpose of the NEST module is to

1. Rebuild a nested structure out of a "flat" graph.
2. Extract morphological information for further analysis.

The graph of the sentence *Ace signs a contract* is

(2) [COMPANY: Ace] ← (AGNT) ← [SIGN] →
(THEME) → [CONTRACT]

This graph is "flat," i.e., word sequence and hierarchical syntactic links between phrases (as represented by a parse tree) are not retained. In order to rebuild an *utterance path*, the system first selects a *root* conceptual relation. This is one of the conceptual relations that correspond to the subject-verb link. The table SINT_SEM described in Section 2 associates the *subject* SP with the following relations:

SUBJ(*vp,*np) ← AGNT(*vp,*np). (e.g., *John eats an apple*),

SUBJ(*vp,*np) ← EXPERIENCER(*vp,*np). (e.g., *John feels cold*),

SUBJ(*vp,*np) ← INSTRUMENT(*vp,*np). (e.g., *The key opens the door*),

SUBJ(*vp,*np) ← PARTICIPANT(*vp,*np). (e.g., *John flies to Rome*),

etc.

The root of graph (2) is the AGNT relation. Once the root relation has been found, the algorithm *explodes* the concepts to the right and to the left of this relation. Given the elementary graph

(3) [C1] ← (ROOT_REL) ← [C2]

exploding a concept C_i means replacing this concept with an elementary graph including it, and iterating the process until all the concepts directly or indirectly linked to C_i have been traversed. For example, if the conceptual graph includes the elementary graph

(4) [C1] ↔ (CONC_REL) ↔ [C3]

after the first step of iteration, (3) becomes

(5) $[[C1] \leftrightarrow (CONC_REL) \leftrightarrow [C3]] \leftarrow$
 $(ROOT_REL) \leftarrow [C2]$

The final form of graph (2) after concept explosion is shown in Figure 9. During this step, semantic information useful to morphological synthesis is extracted and stored for future analysis. Morphosemantic information is conveyed by the concept referent, which determines whether a concept is an individual or a generic instance and whether it points to a unique instance or to a set of concept instances. A second source of morphosemantic data is the tense/mode conceptual relations, such as PAST, PRES, FUT, COND, and POSSIBILITY. These relations are attached to the *main* action of the conceptual graph (i.e., the verb).

Syntactic synthesis

The purpose of the GEN_TREE module is to

1. Replace conceptual relations of the nested graph with syntactic relations.
2. Decide between an active and a passive sentence structure.
3. Create either a simple or a complex sentence structure.
4. Decide concept ordering.

The table SINT_SEM and the grammars of Figures 4 and 6 are used to rebuild a parse tree out of the nested graph of Figure 9.

Several problems are solved during this phase of analysis. First, a passive or an active structure is selected by the user. (This might seem a little naive, but our objective has simply been to provide the system with the ability to generate passive constructions. Given the limited context in which generation is used, producing an active or passive form really does not matter.) If a passive form is selected, the OBJ and the SUBJ links are inverted.

However, neither the SUBJ nor the OBJ links can be found if the graph does not include a concept corresponding to a verb. A graph exemplifying this situation is given in Figure 10. In this case, an *extended* utterance is generated by replacing the *root* conceptual relation with a verb. In the example of Figure 10 the subgraph

$[DELEGATE: \{ \# \}] \leftarrow (PARTICIPANT) \leftarrow [MEETING]$

is replaced by

$[DELEGATE: \{ \# \}] \leftarrow (AGNT) \leftarrow [PARTICIPATE] \rightarrow$
 $(THEME) \rightarrow [MEETING]$

Another commonly encountered problem is that of coordinate constructions. The system generates a coordinate construction whenever a concept is related by the same conceptual relation to different concepts. In Figure 11, for example, the concept DELEGATE is the AGNT of both

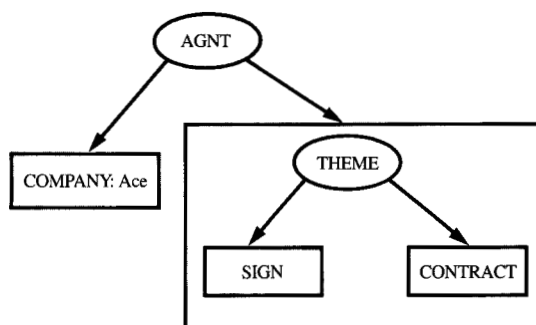


Figure 9

Result of the NEST procedure.

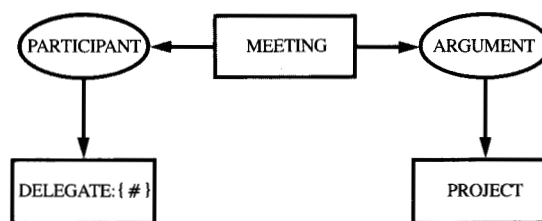


Figure 10

Example of a graph not directly utterable.

SIGN and PASS; a conjunction of VPs is hence created: *The delegates (sign a contract) and (approve a project).*

The GEN_TREE algorithm also decides on word order when this is not forced by rules of the grammar. In the above example, the two coordinate VPs could have been uttered in the opposite order. Word order is also decided for adjectives, which in Italian may appear either before or after the noun they modify.

At the end of this step, the GEN_TREE produces a parse tree of the type of Figure 5. The terminal nodes are, however, still *concepts*. Replacement of concepts by words is performed by MORPH.

Morphological synthesis

The purpose of the MORPH module is to replace concepts with words and to introduce determiners. Replacing

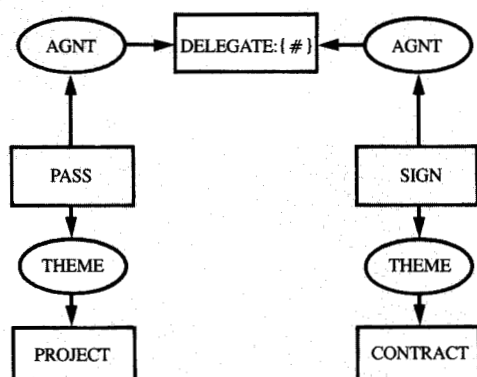


Figure 11

Example of a graph that can be uttered with a coordinate construction.

concepts with words implies selecting among synonyms and near-synonyms of a given word-sense concept. This information is contained in a table associating words with concept names. For example, the concept PROJECT corresponds to the words *plan* and *project*; the concept AGREEMENT corresponds to the word *agreement* and to its near-synonyms *pact*, *arrangement*, and *compromise*. Words are selected at random. The synonym table gives for each concept name a list of *stems*; in order to restore morphological information and attach determiners to nouns, the following rules are applied:

1. *Nouns*. Get the number from the concept referent.
2. *Verbs*. Get the number from the subject and the tense from the time/mode conceptual relations (PAST, PRES, FUT, COND, etc.) extracted by the NEST module.
3. *Adjectives*. Get the gender and number from the noun (or nouns) modified by the adjective.
4. *Determiners*. Use a definite or indefinite article if the referent of the concept associated with the noun is a constant or a variable, respectively; get the gender and number from the noun (in Italian determiners are inflected).

The stem and the morphological data derived by these rules are the input conditions for morphological synthesis. Words are decomposed into prefix, stem, and suffix, and the appropriate word ending is derived from a table of word-ending models. Exceptions to standard word-ending models are also handled [15].

The result of this step is an NL answer to a query or a paraphrase of a previously analyzed text. For example, the graph of Figure 10 gives rise to the following utterances:

*The delegates participate in a meeting on a project,
A project is the theme of a meeting of the delegates,
A plan is the theme of an assembly of the delegates,
etc.*

4. Concluding remarks and future developments

This paper has described the language-analysis and -generation modules of a text processor developed at the IBM Scientific Center in Rome.

The kernel of the system is a *conceptual lexicon* where detailed knowledge on *word-sense uses* (surface semantic patterns) is stored. The knowledge-representation model is that of conceptual graphs.

A prototype of the system has been implemented and provided with graphic features to show intermediate steps of analysis (morphological decomposition of words, parse trees, and conceptual graphs). The system coverage decreases from morphology to semantics: The morphological processor is able to analyze 100% of the words included in the natural corpus; the syntactic processor covers about 80% of the sentences; and, finally, the semantic analyzer processes morphologically and syntactically parsed sentences, provided the words are included in the conceptual lexicon. Within the limit imposed by the size of the lexicon, complex language phenomena such as metonymy, idiomatic expressions, and multiple prepositional ambiguities are handled.

In our project, we have made an effort toward defining a rich semantic lexicon. Currently, the conceptual lexicon includes some 850 detailed word-sense definitions, each including an average of 20 surface semantic patterns (SSPs). The advantage of SSPs is twofold: First, they account for many language phenomena commonly found in any unrestricted NL domain; second, they are acquired inductively, by looking at all the occurrences of a given word within the natural corpus. When a new aspect of a defined word-sense arises at a later time (e.g., some new idiomatic expression or less common word pattern), it can simply be added to the definition. A third advantage, admittedly controversial, is that our semantic knowledge framework does *not* aim to express the *real meaning* of a word; rather, it *describes* the uses of a word, as found in the language. The problem of word decomposition into "deeper" meaning elements, with or without primitives, is one that has preoccupied philosophers and linguists since the early history of these disciplines. We have deliberately avoided this "black hole"; less ambitiously, we have decided to provide our program with some knowledge of surface language mechanisms. Paraphrasing Wilks [9], some of the systems that "claim to be 'superficial' have far more features in common with the 'deep' ones than one might expect, if one

examines their target parsing structures." (In the paper by Wilks, the words "superficial" and "deep" were exchanged.)

One could object that these goals are nevertheless very ambitious, given the enormous number of patterns found in language. The good news is that language patterns can be encoded easily by looking at existing texts; a concept editor for simplifying knowledge entry has already been developed. A new project was recently started to build a tool for SSP acquisition. First, the syntactic parser detects valid word associations by analyzing all the sentences in which the word to be defined is found. The syntactically valid word patterns are used to generate, by generalization and analogy, a set of SSPs for a word. Three types of information sources are used: i) correspondence rules between syntactic links and conceptual relations (see the section on the semantic interpretation algorithm); ii) knowledge about the conceptual category belonging to a word (e.g., *contract* is a DOCUMENT); iii) rules about the use of conceptual relations (e.g., the selectional restriction imposed by POSSESS is [ANIMATE] → (POSSESS) → [CONCRETE]). Other heuristics are being investigated. The tool is semiautomatic, i.e., new SSPs must be accepted by a human expert before their insertion in the knowledge base.

In conclusion, we believe that a wider range of models and experimentation should be pursued in the field of lexical knowledge acquisition. This might involve large and perhaps even frustrating efforts, but no short-cut solutions are available to perform the very complex task of language manipulation by computers.

5. Appendix

This appendix provides an example of concept definition. The amount of detail given in each word-sense definition is considerably more than in Figure 3; on average, each definition includes about 20 elementary graphs. Graphs are implemented in logic by rules of the form

```
CONC_REL(C,*Y) ← COND(Y,*Y).
CONC_REL(*Y,C) ← COND(Y,*Y).
```

which are read, respectively, as "**Y* modifies the concept *C* by the relation *CONC_REL* if the condition *COND* can be proved" and "**Y* is modified by the concept *C* through the relation *CONC_REL* if the condition *COND* can be proved." The first argument of *COND* is either an abstract concept type or a list of word-senses; notice, however, that the predicate *COND* requires more complex computation than a simple type test [13].

The example provided hereafter is a translation of the Italian word *accordo* ('agreement'). Because of the translation, a few rules handling specific Italian idioms were omitted. On the other hand, the translation possibly lacks some commonly encountered uses of the English word *agreement*. In any case, it should be understood that the overall objective of this work is not to contribute to the field

of linguistics, but rather to show the viability of the SSP approach.

Each rule is preceded by a statement *S* that gives its "reading," and is followed by an example. Note also that some rules might include exceptions (for example, the verb *run* is a kind of POSITION_MODIFY_ACT, but the expression "to run an agreement" is unacceptable). It is the semantic-verification algorithm that detects and eliminates these inconsistencies by double-checking the truth of a condition.

Hence, for example, *CONC_REL(x,y)* is true *iff*

```
CONC_REL(x,y) ← (x: COND(Y,y)) & (y: COND(X,x)).
```

The key idea is to write tighter conditions whenever these can be stated easily, using the available conceptual categories. If a rule has an exception, it is the task of semantic verification to detect it by means of the double-check.

AGREEMENT:

supertype(PARTICIPATION_ACT, AGREEMENT).

Statement 1 The actors of an agreement are HUMAN entities (individuals or organizations)

```
PARTICIPANT(AGREEMENT.*referent.nil,*concept_name)
← cond(HUMAN,*concept_name).
```

Example *an agreement between Italy and France; the agreement of delegates*

Statement 2 The argument of an agreement might be whatever (TOP)

```
ARGUMENT(AGREEMENT.*referent.nil,*concept_name)
← cond(TOP,*concept_name).
```

Example *an agreement on new taxes; an agreement about social assistance*

Statement 3 The purpose of an agreement is MENTAL, SOCIAL, and MODIFICATION acts

```
PURPOSE(AGREEMENT.*referent.nil,*concept_name)
← cond(MENTAL_ACT,*concept_name).
```

Example *an agreement to impose new taxes*

```
PURPOSE(AGREEMENT.*referent.nil,*concept_name)
← cond(SOCIAL_ACT,*concept_name).
```

Example *an agreement for the election of new delegates*

```
PURPOSE(AGREEMENT.*referent.nil,*concept_name)
← cond(COLLABORATION.SUPPLY.COOPERATION.nil,
*concept_name).
```

```
PURPOSE(AGREEMENT.*referent.nil,*concept_name)
← cond(SITUATION_MODIFY_ACT,*concept_name).
```

Example an agreement to restore/change the structure of a company

Statement 4 DESCRIPTIVE, MODIFICATION, and SOCIAL acts have as a theme an agreement

THEME(*concept_name,AGREEMENT.*referent.nil)
← cond(DESCRIPTIVE_ACT,*concept_name).

Example to illustrate the agreement; to describe the agreement

THEME(*concept_name,AGREEMENT.*referent.nil)
← cond(SOCIAL_ACT,*concept_name).

Example to sign an agreement; the stipulation of an agreement

THEME(*concept_name,AGREEMENT.*referent.nil)
← cond(SITUATION_MODIFY_ACT,*concept_name).

Example to modify the agreement; to conclude an agreement

THEME(*concept_name,AGREEMENT.*referent.nil)
← cond(POSITION_MODIFY_ACT,*concept_name).

Example to get to an agreement; to reach an agreement; to come to an agreement

Statement 5 An agreement has a TEMPORAL location

TIME(AGREEMENT.*referent.nil,*concept_name)
← cond(TIME_ENTITY,*concept_name).

Example a three-month agreement; the agreement of October

Statement 6 An agreement has MODAL/MANNER qualifiers

MANNER(AGREEMENT.*referent.nil,*concept_name)
← cond(ACT_QUALITY,*concept_name).

Example a good/interesting/relevant agreement

MANNER(AGREEMENT.*referent.nil,*concept_name)
← cond(MODAL_CIRCUMSTANCE,*concept_name).

Example the effectiveness of the agreement

Statement 7 An agreement takes place in some LOCATION

PLACE(AGREEMENT.*referent.nil,*concept_name)
← cond(LOCATION,*concept_name).

Example the agreement at Rome

Statement 8 An agreement might cause an advantage or disadvantage to HUMAN ENTITIES

AGAINST(AGREEMENT.*referent.nil,*concept_name)
← cond(HUMAN_ENTITY,*concept_name).

Example an agreement against the company

ADVANTAGE(AGREEMENT.*referent.nil,*concept_name)
← cond(HUMAN_ENTITY,*concept_name).

Example an agreement in favor of Ace

References

1. John F. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley Publishing Co., Reading, MA, 1984.
2. J. Slocum, "Machine Translation: An American Perspective," *Proc. IEEE* 74, No. 7, 969-978 (July 1986).
3. D. Lenat and E. Feigenbaum, "On the Thresholds of Knowledge," *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, Milan, 1987, pp. 1173-1182.
4. J. L. Binot and K. Jensen, "A Semantic Expert Using an On-Line Standard Dictionary," *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, Milan, 1987, pp. 709-714.
5. K. Dahlgren and J. McDowell, "Kind Types in Knowledge Representation," *Proc. COLING '86 (Proceedings of the 11th International Conference on Computational Linguistics)*, Bonn, West Germany, August 25-29, 1986, pp. 216-221.
6. K. Dahlgren and J. McDowell, "Commonsense Reasoning with Verbs," *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, Milan, 1987, pp. 446-448.
7. G. E. Heidorn, "Augmented Phrase Structure Grammar," *Theoretical Issues in Natural Language Processing*, R. Shank and B. Nash-Webber, Eds., Association for Computational Linguistics, 1975.
8. U. Zernik, "Strategies in Language Acquisition: Learning Phrases from Examples in Contexts," Ph.D. Dissertation, *Technical Report No. UCLA-AI-87-1*, University of California, Los Angeles, 1987.
9. Y. Wilks, "Deep and Superficial Parsing," *Parsing Natural Language*, M. King, Ed., Academic Press, Inc., New York, 1983, pp. 219-246.
10. F. Antonacci, M. T. Pazienza, M. Russo, and P. Velardi, "A System for the Analysis and Generation of Sentences in Italian," *Conceptual Structures for Knowledge Representation*, J. Sowa, N. Foo, and A. Rao, Eds., to be published.
11. F. Antonacci, "Using Semantic Hints in a Syntactic Parser," *IBM Report No. SCR-0004*, Rome Scientific Center, May 1987.
12. M. T. Pazienza and P. Velardi, "A Structured Representation of Word Senses for Semantic Analysis," *3rd Conference of the European Chapter of the ACL*, Copenhagen, April 1-3, 1987, pp. 249-257.
13. M. T. Pazienza and P. Velardi, "Integrating Conceptual Graphs and Logic in a Natural Language Understanding System," *Second International Workshop on Natural Language Understanding and Logic Programming*, Vancouver, B.C., August 17-19, 1987, pp. 138-152.
14. M. T. Pazienza and P. Velardi, "Representing Word Senses for Semantic Analysis," *Conceptual Structures for Knowledge Representation*, J. Sowa, N. Foo, and A. Rao, Eds., to be published.
15. M. Russo, "A Generative Grammar Approach for the Morphologic and Morphosyntactic Analysis of Italian," *3rd Conference of the European Chapter of the ACL*, Copenhagen, April 1-3, 1987, pp. 32-37.
16. M. Russo, "A Rule Based System for the Morphologic and Morphosyntactic Analysis of the Italian Language," *Second International Workshop on Natural Language Understanding and Logic Programming*, Vancouver, B.C., August 17-19, 1987, pp. 185-196.
17. T. Winograd, *Understanding Natural Language*, Edinburgh University Press, Scotland, 1972.
18. R. Wilensky, "A Knowledge-Based Approach to Language Processing: A Progress Report," *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, 1981, pp. 25-30.

19. P. Jacobs, "Generation in a Natural Language Interface," *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, 1983, pp. 610-612.
20. G. Ritchie, "Semantics in Parsing," *Parsing Natural Language*, M. King, Ed., Academic Press, Inc., New York, 1983, pp. 199-218.
21. J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley Publishing Co., Reading, MA, 1979, pp. 248-269.
22. John F. Sowa, "Using a Lexicon of Conceptual Graphs in a Semantic Interpreter," *Relational Models of the Lexicon*, Martha W. Evens, Ed., Cambridge University Press, to appear.
23. Y. Wilks, "Preference Semantics," *Formal Semantics of Natural Language*, Cambridge University Press, 1975.
24. R. C. Schank, "Conceptual Dependency: A Theory of Natural Language Understanding," *Cogn. Psychol.* 3 (1972).
25. R. M. Weischedel, "Knowledge Representation and Natural Language Processing," *Proc. IEEE* 74, No. 7, 905-920 (July 1986).
26. C. Fillmore, "The Case for Case," *Universals in Linguistic Theory*, E. Bach and R. Harms, Eds., Holt, Rinehart & Winston, New York, 1968, pp. 1-81.
27. D. McDonald, "Natural Language Generation: Complexities and Techniques," *Machine Translation*, S. Nirenburg, Ed., Cambridge University Press, 1987, pp. 192-244.
28. K. McKeown, "Language Generation: Application, Issues, and Approaches," *Proc. IEEE* 74, No. 7, 961-968 (July 1986).
29. N. G. Goldman, "Conceptual Generation," *Conceptual Information Processing*, R. Schank, Ed., North Holland/American Elsevier, New York, 1975, pp. 289-371.
30. K. McKeown, "Using Focus to Constrain Language Generation," *Computational Models of Natural Language Processing*, B. Bara and G. Guida, Eds., Elsevier Science Publishers, New York, 1984, pp. 261-273.

Received March 6, 1987; accepted for publication October 20, 1987

Paola Velardi *IBM Italy, Scientific Center, Via Giorgione 159, 00147 Rome, Italy.* Professor Velardi received her degree in electrical engineering in 1978 from the University "La Sapienza" of Rome. She has been a researcher with the Ugo Bordoni Foundation in Rome and with the Center for Reliable Computing at Stanford University, California. Since 1984, she has been with the IBM Scientific Center in Rome. This year Professor Velardi received an associate professorship at the University of Ancona. Her current research interests are knowledge representation and natural-language processing.

Maria Teresa Pazienza *University "La Sapienza," Department of Computer Science and Systems, Via Buonarroti 12, 00185 Rome, Italy.* Professor Pazienza received her degree in physics in 1972 from the University of Bari. She has worked in cooperation with such European research centers as CCR Euratom and ESA-ESRIN. Since 1985, she has been an associate professor in the engineering faculty of the University "La Sapienza" of Rome. Professor Pazienza's main research interests are databases, office automation, and natural-language processing.

Mario De' Giovanetti *viale Roma 47, 28100 Novara, Italy.* Mr. De' Giovanetti received his degree in electrical engineering from the Polytechnic Institute of Milan in 1987. He developed a thesis on natural-language generation at the IBM Scientific Center in Rome.